Electronic Letters on Computer Vision and Image Analysis 7(2):76-95, 2008

Thinning-free Polygonal Approximation of Thick Digital Curves Using Cellular Envelope

Partha Bhowmick*, Arindam Biswas*, and Bhargab B. Bhattacharya+

* *Computer Science and Technology Department*
Bengal Engineering and Science University, Shibpur, Howrah, India
emails: bhowmick@gmail.com, abiswas@cs.becs.ac.in

+ *Advanced Computing and Microelectronics Unit*
Indian Statistical Institute, Kolkata, India
email: bhargab@isical.ac.in

Received 15th May 2008; accepted 11th July 2008

Abstract

Since the inception of successful rasterization of curves and objects in the digital space, several algorithms have been proposed for approximating a given digital curve. All these algorithms, however, resort to thinning as preprocessing before approximating a digital curve with changing thickness. Described in this paper is a novel thinning-free algorithm for polygonal approximation of an arbitrarily thick digital curve, using the concept of "cellular envelope", which is newly introduced in this paper. The cellular envelope, defined as the smallest set of cells containing the given curve, and hence bounded by two tightest (inner and outer) isothetic polygons, is constructed using a combinatorial technique. This envelope, in turn, is analyzed to determine a polygonal approximation of the curve as a sequence of cells using certain attributes of digital straightness. Since a real-world curve/curve-shaped object with varying thickness, unexpected disconnectedness, noisy information, etc., is unsuitable for the existing algorithms on polygonal approximation, the curve is encapsulated by the cellular envelope to enable the polygonal approximation. Owing to the implicit Euclidean-free metrics and combinatorial properties prevailing in the cellular plane, implementation of the proposed algorithm involves primitive integer operations only, leading to fast execution of the algorithm. Experimental results that include output polygons for different values of the approximation parameter corresponding to several real-world digital curves, a couple of measures on the quality of approximation, comparative results related with two other well-referred algorithms, and CPU times, have been presented to demonstrate the elegance and efficacy of the proposed algorithm.

1 Introduction

The subject on properties, characterizations, and representations of digital curves (DC) has been researched continuously since the debut of digitization of graphical objects and visual imageries [28, 29, 34]. Nevertheless, in the abundance of various problems and their algorithms related with digital objects, polygonal approximation of a digital curve/object has received special attention for its efficient representation and for its potential applications in connection with analysis of digital images [1, 5, 25]. For, the set of straight edges of the concerned polygon carries a strong geometric property of the underlying objects, which can be used for efficient high level description of the objects and for finding the similarity among different objects in the digital plane.

Correspondence to: <pb@cse.iitkgp.ernet.in; bhowmick@gmail.com>
A preliminary version of this paper appeared in [10].

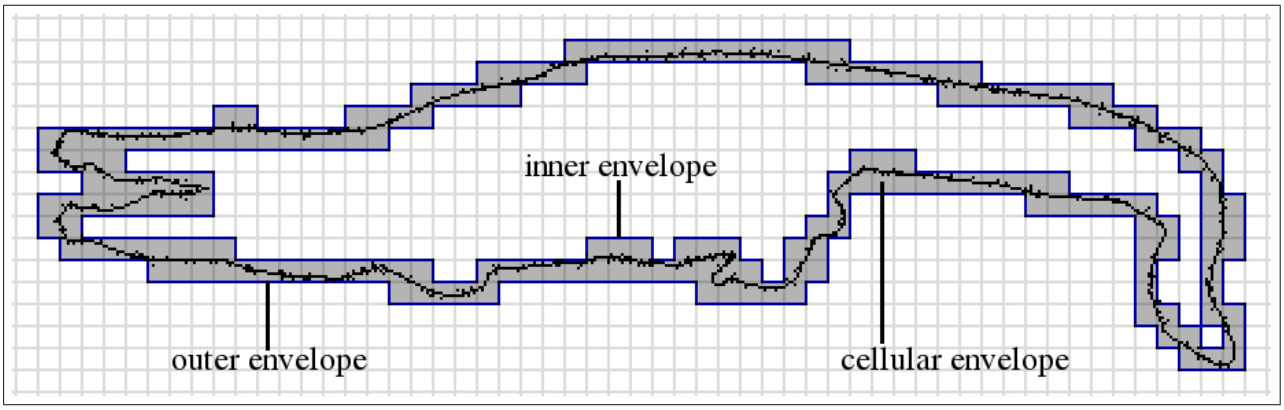


Figure 1: Cellular envelope $\mathcal{E}(\mathcal{C}, \mathcal{G})$ of a real-world (thick, rough, and reducible) curve-shaped object $\mathcal{C}$ for cell size $g = 8$.

Since an optimal solution of polygonal approximation targeted to minimize the number of vertices, and space thereof, is computationally intensive, several heuristic and meta-heuristic approaches based on certain optimality criterion have been proposed over the last few decades, and some of these that have come up in recent times may be seen in [8, 31, 38, 39, 40, 41, 47]. Further, there also exist various studies and comparisons of the proposed techniques, e.g., [8, 35, 41, 46], to cite a few. This entire collection of polygonal approximation algorithms, however, consider the input digital curve to be strictly "irreducible"* (and connected thereof), failing which the algorithm may produce undesired results pertaining to polygonal approximation.

Hence, in case of a thick DC, thinning is required to ensure the property of "irreducibility" to the input DC so that it can qualify for the subsequent process of polygonal approximation. A thinning procedure, being plagued by asymmetric erosion in the thick regions and shifting of junction/end points, and being liable to slow down the overall run time of the approximation process, is susceptible to deteriorate the results of approximation. Furthermore, the result goes on worsening if there occurs some missing grid points (pixels) in the input DC — which splits, therefore, into multiple DC's — producing several approximate polygons instead of a single polygon, thereby giving rise to misleading impression, and more specifically, posing severe problems in the subsequent applications. These problems have been tackled in our method using the novel concept of cellular envelope of an arbitrary digital curve whose thickness may vary non-uniformly. In our method, we consider that all possible thicknesses — including 0 (missing pixel) and 1 (one pixel thick) — may occur in a (connected or disconnected) DC (or a curve-shaped object)† when it is subject to polygonal approximation.

A brief outline of the paper is as follows. In Sec. 2, we present a combinatorial algorithm to derive the cellular envelope of an arbitrary DC (stage I) using its inner and outer isothetic polygons [7, 11, 49]. Sec. 3 enumerates some digital geometric properties of cellular line segments (CSS), followed by the motivation and underlying principle for their extraction (stage II) from the cellular envelope of the input DC obtained in stage I. In Sec. 4, we present our method **PACE** with the major steps of the two algorithms corresponding to stage I and stage II, along with their brief analysis. Sec. 5 exhibits some test results on some typical curve-shaped objects with varying curve thickness, which shows the elegance and efficiency of the proposed method. Finally in Sec. 6, we summarize the strength of our method, comment on its future scope and further works, and point out the possibilities of polygonal approximation in the cellular plane.

*A digital curve $\mathcal{C}$ is said to be "irreducible" if and only if removal of any grid point p in $\mathcal{C}$ makes $\mathcal{C}$ disconnected.

†In this paper, we use the term "DC" to denote a digital curve (reducible or irreducible) as well as a curve-shaped object that may contain multiple disconnected segments producing the impression of a single object.

1.1 Existing Methods

Algorithms for approximating a given digital curve or contour, which is one-pixel thick, had been proposed since the early period of digitization [1, 5, 25]. Efficient and suboptimal algorithms of several variants had been proposed later [9, 10, 11, 31, 38, 39]. The problem of polygonal approximation of a digital curve persisted to be engrossing even today, and with the inset of new paradigms that opens up new possibilities, a number of algorithms have been proposed in recent times [4, 3, 12, 13, 22, 45]. All these methods are meant for polygonal approximation of a strictly one-pixel thick digital curve and mostly based on the conventional techniques, such as an appropriate distance criteria, usage of masks, eigenvalue analysis, Hough transform, etc. In general, all these algorithms can be broadly classified into two categories — one in which the number of vertices of the approximate polygon(s) is specified, and the other where a distortion criterion (e.g., maximum Euclidian distance) is used. The principles and salient features of some of these algorithms are mentioned below.

Amongst the earlier algorithms, the scan-along algorithm proposed by Wall and Danielsson [42] needs a mention here for its high speed of execution owing to a simple-yet-effective concept of *area deviation* for each line segment. The algorithm outputs a new line segment when the area deviation per unit length of the current approximating segment exceeds a prescribed threshold. Provision for simplifying the associated computations is there, e.g., by replacing $\sqrt{x^2 + y^2}$ with $(|x| + |y|)$ or $\max\{|x|, |y|\}$. However, the algorithm lacks the scope of producing the desired result for (self-)intersecting or branching curves when the input set of points constituting the curve-set is not given in the order that defines the curve.

Another procedure proposed by Teh and Chin [41] detects the *dominant points* in a closed digital curve, given in the chain-coded representation as input. A dominant point corresponds to *curvature maxima* in the local neighborhood of the concerned curve. The procedure first determines the *region of support* for each point based on its local properties, computes measures of relative significance (e.g., cosine curvature, k curvature, 1 curvature, etc.) of each point, and finally detects dominant points by *non-maxima suppression*. However, since the procedure resorts to trigonometric functions for curvature finding and performs non-maxima suppression in multiple iterations (4 passes), its runtime is quite high.

Later, the concept of *perceptual organization* was introduced by Hu and Yan [24], which attempts to match the human performance. The approximation process is divided into three stages. In the first stage, points are grouped together using a *linking-merging* approach based on the principles of proximity, similarity, and symmetry. In the second stage, the linked curve is smoothed so as to suppress noise and delete visually insignificant points. Finally, a rule-based strategy is applied to preserve the feature points while reducing the number of segments in the approximated polygon. The procedural complexity and the runtime are high due to three stages, each using multiplications for the entire set of points on the curve.

Another method of polygonal approximation using *ant colony search* technique is proposed by Yin [47]. A directed graph is used to represent the problem with the objective of finding the *shortest closed circuit* on the graph under the problem-specific constraints. A number of artificial ants are distributed on the graph, which communicate with one another through the *pheromone trails* as a long-term memory guiding the future exploration of the graph through certain *node transition rules* and *pheromone updating rules*. The procedural complexity is very high due to its inherent recursive nature and complex calculations, one of which is exponentiation in selection problem of a node.

A comparative study of the above algorithms can be found in some of the recent papers [8, 46]. Since most of these algorithms require intensive floating-point operations in order to analyze the discrete curvature [2, 17, 20, 41, 44], their runtime for a complex digital curve is quite high. For other details, the related procedures in several other works [6, 15, 30, 42, 43, 35, 41, 47, 48] may be looked at. Further, for thick, rough, and weakly disconnected digital curves (which are usually representatives of the corresponding real-world objects), the algorithms are susceptible to produce unexpected results. On the contrary, the algorithm proposed by us can deal with such non-ideal curves, and and yields a suboptimal solution of polygonal approximation uses primitive integer operations only.

2 Cellular Envelope

If $\mathcal{C}$ be a given DC, and $\mathcal{G} = (\mathcal{H}, \mathcal{V}, g)$ be a set of uniformly spaced horizontal grid lines ($\mathcal{H}$) and vertical grid lines ($\mathcal{V}$) with spacing g, then the cellular envelope of $\mathcal{C}$, corresponding to the cellular plane defined by $\mathcal{G}$, is given by

$$\begin{aligned} \mathcal{E}(\mathcal{C}, \mathcal{G}) &= \mathcal{E}_{out}(\mathcal{C}, \mathcal{G}) \setminus \mathcal{E}_{in}(\mathcal{C}, \mathcal{G}) \text{ if } \mathcal{C} \text{ is closed or contains a closed part,} \\ &= \mathcal{E}_{out}(\mathcal{C}, \mathcal{G}) \text{ if } \mathcal{C} \text{ is open,} \end{aligned} \tag{1}$$

where $\mathcal{E}_{out}(\mathcal{C}, \mathcal{G})$ and $\mathcal{E}_{in}(\mathcal{C}, \mathcal{G})$ represent the respective outer and inner envelopes corresponding to $\mathcal{C}$ w.r.t. $\mathcal{G}$, such that

– each point $p \in \mathcal{C}$ should lie inside $\mathcal{E}_{out}(\mathcal{C}, \mathcal{G})$ and outside $\mathcal{E}_{in}(\mathcal{C}, \mathcal{G})$;
– each vertex of $\mathcal{E}(\mathcal{C}, \mathcal{G})$ (and of $\mathcal{E}_{out}(\mathcal{C}, \mathcal{G})$ and $\mathcal{E}_{in}(\mathcal{C}, \mathcal{G})$, thereof) is in $\mathcal{G}$;
– area of $\mathcal{E}(\mathcal{C}, \mathcal{G})$ is minimized.

The cellular envelope of a DC (curve-shaped object) $\mathcal{C}$, which is rough, not irreducible, and disconnected (since it has uneven thickness and stray pixels) has been shown in Fig. 1. Note that, the cellular envelope $\mathcal{E}(\mathcal{C}, \mathcal{G})$ shown in this figure is for cell size $g = 8$, and the envelope "tightly encloses" all the points of $\mathcal{C}$ with no points lying outside $\mathcal{E}(\mathcal{C}, \mathcal{G})$.

2.1 Combinatorial Properties of a Cellular Envelope

Let $\mathcal{I}$ be the 2D image plane having height h and width w, and containing the entire object $\mathcal{C}$. Let $\alpha(i, j)$ be the point of intersection of the horizontal grid line $l_H : x = i \in \mathcal{H}$ and the vertical grid line $l_V : y = j \in \mathcal{V}$. Let $\mathcal{S}_{\rm LT}$, $\mathcal{S}_{\rm RT}$, $\mathcal{S}_{\rm LB}$, and $\mathcal{S}_{\rm RB}$ be the respective left-top, right-top, left-bottom, and right-bottom square cells with the common grid point $\alpha(i, j)$, and let $\alpha'(i, j + g)$ and $\alpha''(i + g, j)$ be the respective grid points lying immediate right and lying immediate below α, as shown in Fig. 2.

We construct a binary matrix A_e (*edge matrix*) that contains $\big((w/g)(h/g+1)\big) \times \big((h/g)(w/g+1)\big)$ entries, each entry being in one-to-one correspondence with a particular edge of a particular cell. If an edge $e(\alpha, \beta)$ connecting two neighbor grid points α and β is intersected by the object $\mathcal{C}$, then the corresponding entry in A_e is '1', otherwise '0'.

Now, from A_e, we construct another binary matrix A_c (*cell matrix*) of size $(h/g) \times (w/g)$, in which each entry corresponds to a unique cell — the entry is '1' if at least one of the four edges of the concerned cell is intersected by the object $\mathcal{C}$, and is '0' otherwise — which is checked from the correspondence of its edge information in A_e.

Next, the candidature of α as a vertex of the (inner or outer) envelope is checked by looking at the combinatorial arrangements (w.r.t. object containments) of the four cells having common vertex α. There exist $2^4 = 16$ different arrangements of these 4 cells, since each cell has 2 possibilities ('0'/'1'). These 16 arrangements can be further reduced to 5 cases, where, a particular case $\mathbf{C}_q$, $q = 0, 1, \ldots, 4$, includes all the arrangements where

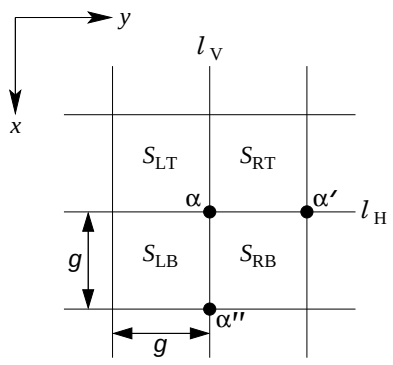


Figure 2: Four cells with common vertex α.

exactly q out of these 4 squares has/have object containments (i.e., contain(s) parts of the object $\mathcal{C}$), and the remaining (i.e., $4-q$) ones have not. That is, the case in which the sum of the 4 bits in the corresponding entries in A_c is equal to q is represented by $\mathbf{C}_q$. Further, out of these 5 cases, cases $\mathbf{C}_1$ and $\mathbf{C}_3$ *always* and case $\mathbf{C}_2$ *conditionally* produce vertices of the inner/outer envelope, as explained below (also see Fig. 3).

Case $\mathbf{C}_1$. $\binom{4}{1} = 4$ arrangements (one shown in Fig. 3) are possible where only one cell with vertex α contains $\mathcal{C}$, i.e., exactly one of the corresponding four entries in A_c is '1' and each other is '0'. The envelope will have its one edge ending at α and the next edge starting from α, (shown by the respective arrows in Fig. 3). Hence, if α lies inside $\mathcal{C}$, then it is a 270^0 vertex of $\mathcal{E}_{in}(\mathcal{C},\mathcal{G})$, and if α lies outside $\mathcal{C}$, then it is a 90^0 vertex of $\mathcal{E}_{out}(\mathcal{C},\mathcal{G})$ (the angle $90^0/270^0$ of a vertex means its internal angle in the corresponding envelope).

Case $\mathbf{C}_2$. $\binom{4}{2} = 6$ arrangements (Fig. 3) are possible in which exactly two of the four cells contain $\mathcal{C}$. If the cells containing $\mathcal{C}$ are diagonally opposite (2 out of 6 arrangements), then α is a vertex (90^0 for $\mathcal{E}_{in}(\mathcal{C},\mathcal{G})$ and 270^0 for $\mathcal{E}_{out}(\mathcal{C},\mathcal{G})$); otherwise α is an ordinary point on the envelope perimeter. It should be noticed in Fig. 3 that two different styles of arrow heads indicate the two possibilities of formation of the envelope edges, since α may be visited along either of the two possible paths during traversal, and only the traversed one is required for defining the envelope.

Case $\mathbf{C}_3$. $\binom{4}{3} = 4$ arrangements (Fig. 3) are possible for $q = 3$, where, out of the four cells, only one cell is free. In each such arrangement, α would be a 90^0 vertex for $\mathcal{E}_{in}(\mathcal{C},\mathcal{G})$) and a 270^0 vertex for $\mathcal{E}_{out}(\mathcal{C},\mathcal{G})$).

For case $\mathbf{C}_0$: $\binom{4}{0} = 1$ arrangement, α is just an ordinary grid point lying outside $\mathcal{E}_{out}(\mathcal{C},\mathcal{G})$ or inside $\mathcal{E}_{in}(\mathcal{C},\mathcal{G})$), whereas for case $\mathbf{C}_4$: $\binom{4}{4} = 1$ arrangement, α is a grid point included in $\mathcal{C}$ (since no two traversable edges are incident on it).

3 Cellular Straight Segments

There exist several works on constructs, properties, and applications of cell complexes and cellular straight segments (CSS) in which the primal as well as many alternative definitions of CSS are found [16, 21, 26, 29, 27]. For example, as indicated in [29], a CSS $\mathbf{C}$ can be defined as the minimal set of cells c specified by a straight line segment $\mathbf{L} \in \mathbb{R}^2$ such that

$$\mathbf{L} \cap c \neq \emptyset,\ \forall\, c \in \mathbf{C}; \tag{2}$$

$$\text{and} \qquad \mathbf{L} \subset \mathbf{C}, \tag{3}$$

which makes its primal definition.

Another definition of CSS involving the Euclidean metric space is given in [16], in which it has been shown that a cellular curve $\mathbf{C}$ is a CSS if and only if there exists a direction θ and a pair of (parallel) lines in $\mathbb{R}^2$ (tangential to and) containing $\mathbf{C}$, such that the distance between, and measured in the direction (say, $\theta_\perp$) perpendicular to, this pair of lines does not exceed the distance (along $\theta_\perp$) between the closest pair of parallel lines containing the square formed by $(2 \times 2 =)\, 4$ cells sharing a common vertex.

In a recent work [21], an Euclidean-free definition of CSS has been given in terms of "fully partitioned (finite) strings" ($S^{(0)}$) and "higher order derived strings" ($S^{(j)} : j \geq 1$), the latter being derived iteratively from the preceding string (i.e., $S^{(j-1)}$) by replacing the majority symbol substrings of $S^{(j-1)}$ by its length, and by deleting the minority symbols of $S^{(j-1)}$. Subsequently, it has been shown that a string $S\,(= S^{(0)})$ represents a CSS, provided the jth order derived string of S exists for all $j \geq 0$.

Alternatively, in the perspective of digital straightness, if we consider the center points of these edge-connected cells as grid points, then it follows that a family of cells is edge-connected if and only if the set of center points of these cells is 4-connected. Thus CSS provides a suitable option — apart from that provided by digital straight line segments (DSS) [32] — for adjudging the straightness of a curve in the digital plane, as indicated in a contemporary work [29]. A linear off-line algorithm for CSS recognition, based on convex hull construction, is briefly sketched in [26]. In our work, we have designed an on-line algorithm to derive the set of CSS's from the cellular envelope (Sec. 2) of a curve-shaped object, which cannot be subject to direct DSS

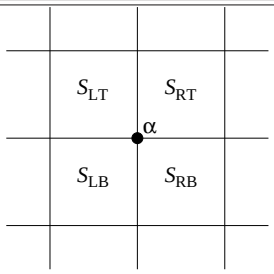
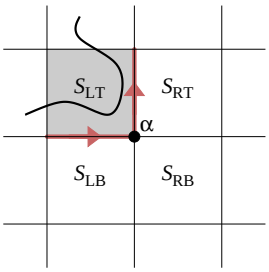
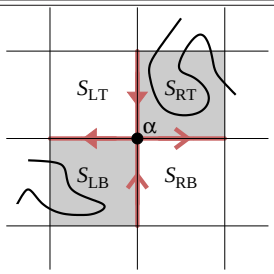
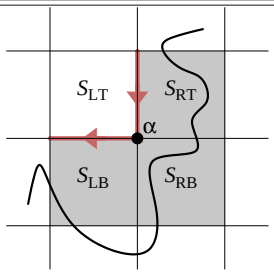
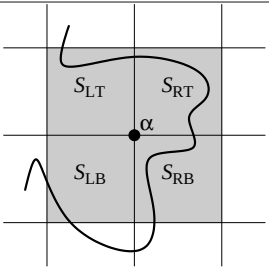
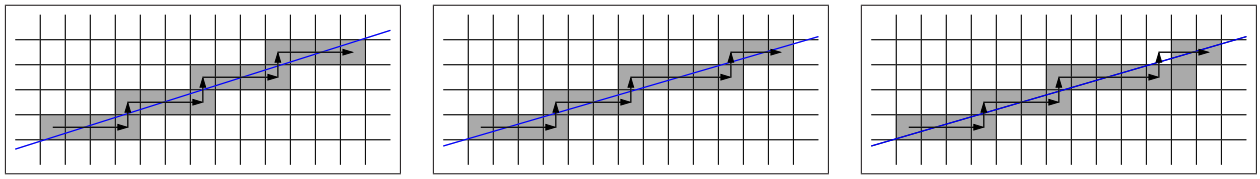
case	no. of arrange-ments/ subcases	an instance with brief explanation	
$\mathbf{C}_0$	1		None of the four cells, namely $\mathcal{S}_{\rm LT}$, $\mathcal{S}_{\rm RT}$, $\mathcal{S}_{\rm LB}$, and $\mathcal{S}_{\rm RB}$, contains any part of $\mathcal{C}$, whereby α is not a vertex of $\mathcal{E}_{in}(\mathcal{C},\mathcal{G})$ or $\mathcal{E}_{out}(\mathcal{C},\mathcal{G})$, and so not a vertex of $\mathcal{E}(\mathcal{C},\mathcal{G})$. Note that, the containment of curve by a cell with vertex α is checked from the cell matrix A_c, which, in turn, is checked from the edge matrix A_e, as explained in Sec. 2.
$\mathbf{C}_1$	4		One of the $(2^2-1=)3$ possible sub-subcases in which $\mathcal{C}$ intersects the left edge and the top edge of $\mathcal{S}_{\rm LT}$ corresponding to the subcase of $\mathcal{S}_{\rm LT}$ containing $\mathcal{C}$. The other two sub-subcases are: one when only the left edge, and one when only the top edge of $\mathcal{S}_{\rm LT}$ is intersected by $\mathcal{C}$. The edges (right edge and bottom edge of $\mathcal{S}_{\rm LT}$), which would belong to either $\mathcal{E}_{in}(\mathcal{C},\mathcal{G})$ or $\mathcal{E}_{out}(\mathcal{C},\mathcal{G})$, have been highlighted and directed to show their directions of traversal with $\mathcal{C}$ lying at the left.
$\mathbf{C}_2$	6		An instance of one of the two possible subcases (out of 6 subcases) for which α is a vertex of $\mathcal{E}_{in}(\mathcal{C},\mathcal{G})$ or $\mathcal{E}_{out}(\mathcal{C},\mathcal{G})$, since here the cells with object containments are diagonally opposite. For each of the other four subcases, the cells would be adjacent to each other with one edge in common, thereby making α as an ordinary edge point of $\mathcal{E}_{in}(\mathcal{C},\mathcal{G})$ or $\mathcal{E}_{out}(\mathcal{C},\mathcal{G})$.
$\mathbf{C}_3$	4		One of the four possible subcases for which α is a 90^0 vertex of $\mathcal{E}_{in}(\mathcal{C},\mathcal{G})$ or a 270^0 vertex of $\mathcal{E}_{out}(\mathcal{C},\mathcal{G})$.
$\mathbf{C}_4$	1		In this case, since α has all the four adjacent cells occupied by the curve, there is no free path (consisting of two edges — one incoming and the other outgoing) through α. Hence, α is neither a vertex of $\mathcal{E}_{in}(\mathcal{C},\mathcal{G})$ nor a vertex of $\mathcal{E}_{out}(\mathcal{C},\mathcal{G})$.

Figure 3: Typical instances of five combinatorial cases. See text for more explanation.



(a) (b) (c)

Figure 4: Examples of cellular curves explaining the significance of straightness properties (R1)–(R4). Note that the directed black path that traces the ordered set of centers of the cells shows the digital curve (DC) corresponding to a cellular curve. The curves in (a) and (b) are CSS's (the corresponding real lines being shown in blue); but the curve in (c) is not, since there does not exist any real line that can pass through the set of cells defining this curve (see text for explanation).

extraction/polygonal approximation due to its inherent nature of possessing varying thickness, as mentioned in Sec. 1.

We have considered the center of each cell for extracting the longest line segment iteratively in (a part of) a cellular envelope $\mathcal{E}(\mathcal{C}, \mathcal{G})$ corresponding to the given curve $\mathcal{C}$ and given cell size g imposed by the grid $\mathcal{G}$. We have used some digital geometric properties of DSS formulated and explained in [29, 32]. Before explaining our algorithm, the DSS properties (defined w.r.t. chain codes [18, 19]) relevant to our work, which were established in [32], and later (see [29]) correlated with the other straightness options such as cellular straightness, are mentioned below[‡].

(R1) The runs have at most two directions, differing by 90^0, and for one of these directions, the run length must be 1.
(R2) The runs can have only two lengths, which are consecutive integers.
(R3) One of the run lengths can occur only once at a time.
(R4) For the run length that occurs in runs, these runs can themselves have only two lengths, which are consecutive integers; and so on.

Few examples of cellular curves/envelopes are shown in Fig. 4 to explain the significance of properties (R1)–(R4). For the curve in (a), if we consider the center of each cell as a grid point, as mentioned earlier, then its chain code is $000200020002000 = 0^3 2 0^3 2 0^3 2 0^3$, which consists of codes 0 and 2 only, and contains consecutive 0's but no two consecutive 2's, thereby satisfying property (R1). Regarding (R2), (R3), and (R4), since there is only one run length (of 0's), this curve trivially satisfies these three properties, and becomes a CSS. Similarly, since the curve in (b) has chain code $0^3 2 0^3 2 0^3 2 0^2$, which obeys (R1)–(R4), it is a CSS. On the contrary, the curve in (c) has chain code $0^3 2 0^3 2 0^5 2 0^1$, which satisfies (R1), but violates (R2) as 0 has non-consecutive run lengths (3 and 5) — even if we do not consider the leftmost and the rightmost run lengths (which are 3 and 1, respectively), and so it is not a CSS.

In our method for extraction of CSS from the cellular envelope $\mathcal{E}(\mathcal{C}, \mathcal{G})$, we have adhered to the properties (R1–R4). In addition, we have considered that also the leftmost and the rightmost run lengths of a CSS should follow property (R2) (which is not mandatory as suggested in [32]).

4 Polygonal Approximation Using Cellular Envelope

The method on finding the (cellular) polygonal approximation of a curve-shaped object $\mathcal{C}$ consists of two stages, namely stage I and stage II. In stage I, we construct the cellular envelope $\mathcal{E}(\mathcal{C}, \mathcal{G})$ based on the novel concept of combinatorial arrangement of the cells containing $\mathcal{C}$, as explained in Sec. 2. In stage II, we analyze the cells of

[‡]In our work, we have considered 4-connectivity of a DSS, i.e., having chain codes lying in the set $\{0, 2, 4, 6\}$, since the cells in the cellular envelope $\mathcal{E}(\mathcal{C}, \mathcal{G})$ obtained for the curve $\mathcal{C}$ (Sec. 2) are connected in 4-neighborhood. In a DSS with 8-connectivity, however, the runs would have directions differing by 45^0 as stated in [32].

STEP 1. Initialize each entry in A_e and each entry in A_c with '0'.
STEP 2. DFS-VISIT on $\mathcal{C}$ starting from p using 8-connectivity to reach the nearest cell edge e_p of $\mathcal{G}$.
STEP 3. DFS-VISIT on A_e starting from the entry $A_e(e_p)$ corresponding to e_p in A_e using 4-connectivity (of '1's in A_e) to assign:
'1' to the entry in A_e corresponding to each cell edge e intersected by $\mathcal{C}$, and
'1' to the entry in A_c corresponding to each of the two cells with e as the common edge.
STEP 4. DFS-VISIT on A_c starting from some cell (e.g., c_p, the left adjacent cell of e_p) of the cellular envelope formed by the '1's obtained in step 3 using 4-connectivity (of '1's in A_c); and check whether the entry $A_c(c)$ corresponding to the cell c currently under DFS-VISIT satisfies at least one of the following two conditions:
(i) both the left and the right adjacent entries of $A_c(c)$ are '1's;
(ii) both the bottom and the top adjacent entries of $A_c(c)$ are '1's.
If (i) or/and (ii) is/are true, then terminate the DFS-VISIT, since the current cell c lies either on a horizontal edge/part (when (i) satisfies) or on a vertical edge/part (when (ii) satisfies) of the cellular envelope of $\mathcal{C}$; and declare c as the seed cell c_0 for stage II.
STEP 5. If no seed cell c_0 is found in step 4, then the cell size is not sufficiently large compared to the (minimum) thickness of the input curve $\mathcal{C}$. Hence the user may be asked to increase the cell size (i.e., grid separation g); alternatively, an arbitrary cell of the envelope may be considered to be the seed cell c_0.

Figure 5: **Algorithm** FIND-CELLULAR-ENVELOPE $(\mathcal{C}, \mathcal{G}, p)$ in stage I.

$\mathcal{E}(\mathcal{C}, \mathcal{G})$ to extract the straight pieces from $\mathcal{E}(\mathcal{C}, \mathcal{G})$, considering the center of each cell of $\mathcal{E}(\mathcal{C}, \mathcal{G})$ as a grid point and using the properties (R1)–(R4), as mentioned and explained in Sec. 3. We have designed and implemented two algorithms, one for each stage, which are briefed up next.

4.1 Stage I: Finding the Cellular Envelope

We consider any point $p \in \mathcal{C}$ as the start point defining the object $\mathcal{C}$. For the time being, consider that $\mathcal{C}$ is connected in 8-neighborhood. Then using DFS-VISIT (Depth First Search algorithm [14]), we can reach the nearest edge e_p of a cell that intersects $\mathcal{C}$. Starting from e_p, using DFS-VISIT on the edges of the cells, we visit those cell edges that are intersected by $\mathcal{E}$; this procedure helps us in constructing the edge matrix A_e and the cell matrix A_c (Sec. 2), which are finally used to obtain $\mathcal{E}(\mathcal{C}, \mathcal{G})$. The major steps of the algorithm FIND-CELLULAR-ENVELOPE $(\mathcal{C}, \mathcal{G}, p)$ to find the cellular envelope of a connected (and of uniform or non-uniform thickness) object $\mathcal{C}$ w.r.t. the cellular array imposed by the grid $\mathcal{G}$ is given in Fig. 5.

In case $\mathcal{C}$ has some missing points/pixels, i.e., possesses disconnectedness, then it may happen that none of the edges of a cell is intersected by $\mathcal{C}$, although $\mathcal{C}$ is contained in that cell. To circumvent this problem, we have to directly construct the cell matrix A_c, without constructing A_e, which would, however, increase the time complexity (and the run time, thereof) of stage I. It may be noted that, if the curve possesses too much gap/disconnectedness, so that the gap is even larger than the cell size, then this gap may result to gap (in the edge-connectivity) of the cells constituting the envelope $\mathcal{E}(\mathcal{C}, \mathcal{G})$, which gets fragmented into two or more pieces, thereby producing faulty results. Choosing an appropriate cell size is, therefore, necessary to obtain the desired cellular envelope of a disconnected DC in stage I.

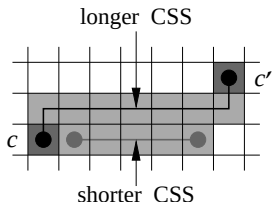
4.2 Stage II: Finding the Cellular Straight Segments

In stage II, the algorithm FIND-CSS $(\mathcal{E}, c_0)$[§], given in Fig. 6, extracts the ordered set of CSS's from the cellular envelope $\mathcal{E}$ as follows. W.l.o.g., since in stage I, the seed cell c_0 lies on a horizontal part (or on a vertical part, or on a thick part) of $\mathcal{E}$, we negotiate two traversals (STEP 1) — one towards left and the other towards right of (center point of) c_0 — to obtain two CSS's with complying (cellular) straightness such that the sum of their

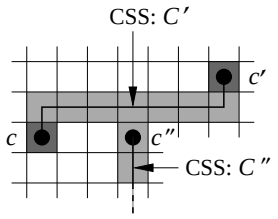
[§]Now onwards, we denote the cellular envelope of $\mathcal{C}$ by $\mathcal{E}$ for simplicity.

STEP 1. Traverse (cell-wise) towards left and towards right from c_0 to extract all possible pairs of CSS starting from c_0, such that
(i) the chain code of each CSS, and
(ii) the combined chain code of the two CSS's
in each pair are in conformity with properties (R1)–(R4);
STEP 2. Find a/the pair of CSS that has maximum sum of lengths;
merge this pair into a single CSS, namely C_1;
declare c_0 and c_1 as the left and the right terminal cells of C_1;
store (the centers of) c_0 and c_1 in the ordered set T.
STEP 3. Start from c_1 to extract the next (longest) CSS, $C_2 := (c_1, c_2)$, with terminal cells c_1 and c_2;
store c_2 in T; and mark the cells defining C_2 as VISITED.
STEP 4. Repeat STEP 3 starting from the last entry (i.e., terminal cell) in T to get the CSS's defining $\mathcal{E}$ until all cells of $\mathcal{E}$ are VISITED (using DFS-VISIT).
Note: (i) If a CSS has both its terminal cells in the 4-neighborhood of another (longer) CSS, then the former (shorter) CSS is not included in T (Fig. 7(a)). (ii) For a bifurcating/branching CSS, we store both its terminal cells in T (Fig. 7(b)).
STEP 5. Declare T as the polygonal approximation of the cellular envelope $\mathcal{E}$.

Figure 6: **Algorithm** FIND-CSS $(\mathcal{E}, c_0)$ in stage II.



(a) A short CSS with each of its terminal cells lying at 4-N of a longer CSS is not considered as a valid CSS (*Note* (i) of STEP 4 in Fig. 6).



(b) For a branching CSS, C'', each of its terminal cells (one is c'' and the other not shown) is stored in T (*Note* (ii) of STEP 4 in Fig. 6).

Figure 7: Inclusion and exclusion of terminal cell(s) of CSS in T.

lengths is maximal, and merge these two to get the first CSS, C_1, to be included in the ordered set T of terminal cells (STEP 2). The starting cell for extracting the next CSS (STEP 3) from the cellular envelope is, therefore, considered to be the right terminal cell c_1 of C_1. We use the algorithm DFS-VISIT [14] to explore the cells constituting the envelope and to extract the CSSs, whose terminal cells are finally reported in T.

Time complexity. If N be the number of points defining the curve $\mathcal{C}$, then its envelope $\mathcal{E}$ consists of $O(N/g)$ cells. Due to DFS-VISITs, therefore, the time complexity in stage I is bounded by $O(N/g)$. In stage II, extraction of each CSS C_i takes $O(|C_i|)$ time, where $|C_i|$ is the number of cells defining C_i. Hence, the time complexity to extract all CSS's in step II is $O\left(\sum |C_i|\right) = O(N/g)$, which gives the total time complexity of **PACE** as $O(N/g)$.

4.3 Efficiency of the Algorithm

The deviation of the approximate polygon(s) (or polychain(s)) from the input set of digital curves determines the efficiency of a polygonal-approximation algorithm. Two such measures to assess the quality of approximation corresponding to a curve $\mathcal{C}$ are (i) *compression ratio* CR $= N/M$ and (ii) *the integral square error* (ISE) between $\mathcal{C}$ and P, N being the number of points in the (thinned) curve $\mathcal{C}$ and M being the number of vertices in the approximate polygon P. It may be noted that there is always a trade-off between CR and ISE [23, 36, 37].

The proposed algorithm is not constrained by the number of vertices of the output polygon. Hence, we cannot use a measure of approximation with M as an input parameter. Since the precision of the cellular envelope $\mathcal{E}$ corresponding to $\mathcal{C}$ depends on the *cell size*, namely g, which, in turn, decides the quality of approximation, the *approximation parameter* of our algorithm is considered as g, depending on which M would change. A high value of g is likely to produce a slacked approximation, whereby M decreases accordingly, whereas a low value of g is expected to output a tight approximation requiring a higher number of vertices to constitute P. Thus, the number of vertices, M, is a measure of the quality of approximation for a given grid size, g. Hence, we measure the efficiency of our algorithm using the compression ratio (CR) versus g.

As stated earlier, only CR does not reflect the overall efficiency of approximation, since a trade-off lies between CR and ISE. Hence, apart from CR, we also find the deviation, $d_\perp(p \to p') := \max\{|x-x'|, |y-y'|\}$, of each point $p(x, y) \in \mathcal{C}$ to its corresponding (nearest) point $p'(x', y') \in P$ for the chosen grid size, g. For all points in $\mathcal{C}$, the overall error of approximation is measured by the frequency $f(d_\perp)$ of the number of points having deviation $d_\perp$ versus $d_\perp$. Further, since $d_\perp$ depends on g in our algorithm, the fraction of the number of points in $\mathcal{C}$ with deviation $d_\perp$ varies with g. Hence, for a given value of g, the *error frequency* is estimated as

$$f(\delta) = \frac{1}{N} \left| \{p \in \mathcal{C} : d_\perp(p \to p') = \delta\} \right|. \tag{4}$$

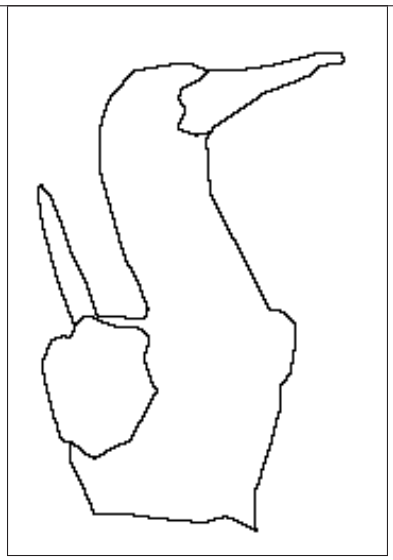
The variation of $f(d_\perp)$ versus $d_\perp$, considering the given value of g, acts as the second measure that provides the error distribution for the polygonal approximation of $\mathcal{C}$. The related plots on (i) CR versus g and (ii) $f(d_\perp)$ versus $d_\perp$ on polygonal approximation of some real-world digital curves are given in Sec. 5.

5 Experiments and Results

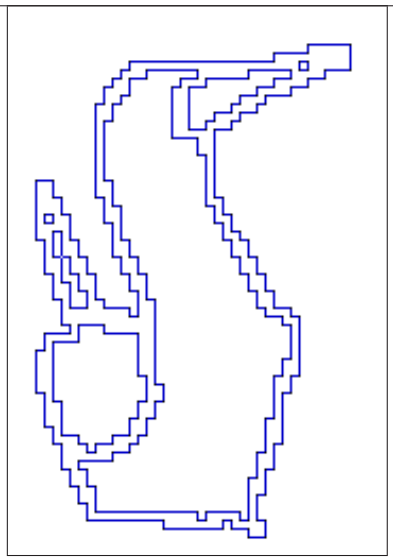
We have implemented the two algorithms, namely FIND-CELLULAR-ENVELOPE and FIND-CSS, that make the proposed method **PACE** for polygonal approximation of an arbitrarily thick DC, in C in SunOS Release 5.7 Generic of Sun_Ultra 5_10, Sparc, 233 MHz. We have tested the algorithms on various digital curves of arbitrary shape, changing thickness, and irregular connectedness, some of which are presented in this paper. It may be mentioned here that, since the concept of a cellular polygon introduced in this work is entirely new, and no other work on cellular polygon exists at present, we could not have a comparative study of our method with any thinning-free method. However, to demonstrate the strength and efficiency of our algorithm, we have given the results of polygonal approximation (on thinned curves) by a couple of existing methods [41, 42]. In order to do this, we have first applied the thinning procedure [33] on a thick curve (or a set of curves) so that the thinned curve can be used as input in these existing algorithms (Fig. 11).

The result for an (non-thinned) edge map of a "duck" is shown in Fig. 8, which testifies the elegance of **PACE** in deriving the cellular polygon corresponding to a DC. It may may be noticed in this figure that, some of the cells in the envelope $\mathcal{E}$ have not been included in any CSS; because in the algorithm FIND-CSS, we have considered the (terminal cells of) each locally longest CSS to be included in P (see the *Note* in STEP 4). But when there is a bifurcation/self-intersection (e.g., in and around the root of its tail) or a sharp bend (e.g., at the tip of its beak), the cellular envelope (Fig. 8(b)) contains several cells across its thickness, which may cause error in the polygonal approximation as manifested in Fig. 8(d) in the part of the polygon corresponding to the region in and around the tail root. Hence a proper value of the cell size, g, is mandatory to ensure a good cellular envelope corresponding to a DC, and a good polygonal approximation thereof.

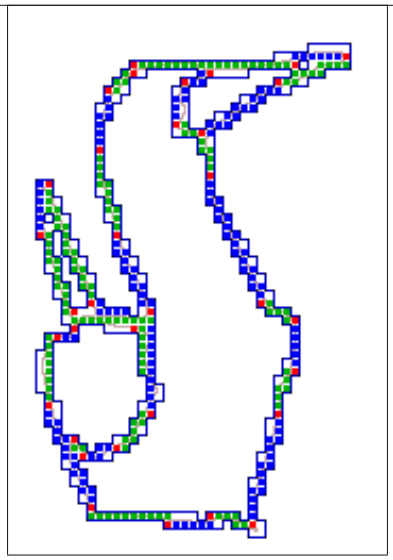
To exhibit the role and significance of g, few other results on the image "duck" have been given in Fig. 9. It is evident from this figure that, for $g = 8$, the approximation deteriorates relative to the one corresponding to $g = 4$ (Fig. 8), owing to the fact that a larger value of g imparts a greater tolerance of approximation by slackening the cellular envelope corresponding to the digital curve. A larger value of g is advantageous when fewer vertices are desired in order to reduce the output complexity, by compromising with a lower quality of approximation. For example, the number of output vertices corresponding to the image "duck" for $g = 8$ is $M(g = 8) = 19$ (Fig. 9), which is significantly less than that corresponding to $g = 4$, i.e., $M(g = 4) = 34$



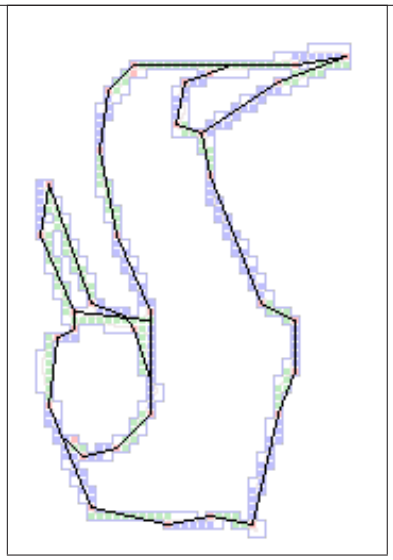
(a) A curve-shaped digital object $\mathcal{C}$ of nonuniform thickness representing the edge map of a "duck" image. Note that, since the curve is not one-pixel thick, the conventional algorithms on polygonal approximation cannot be applied on this curve.



(b) Cellular envelope $\mathcal{E}(\mathcal{C}, \mathcal{G})$ obtained in stage I of the algorithm **PACE**. Note that, the cells of the envelope are connected in 4-neighborhood, which are, therefore, 4-cells.



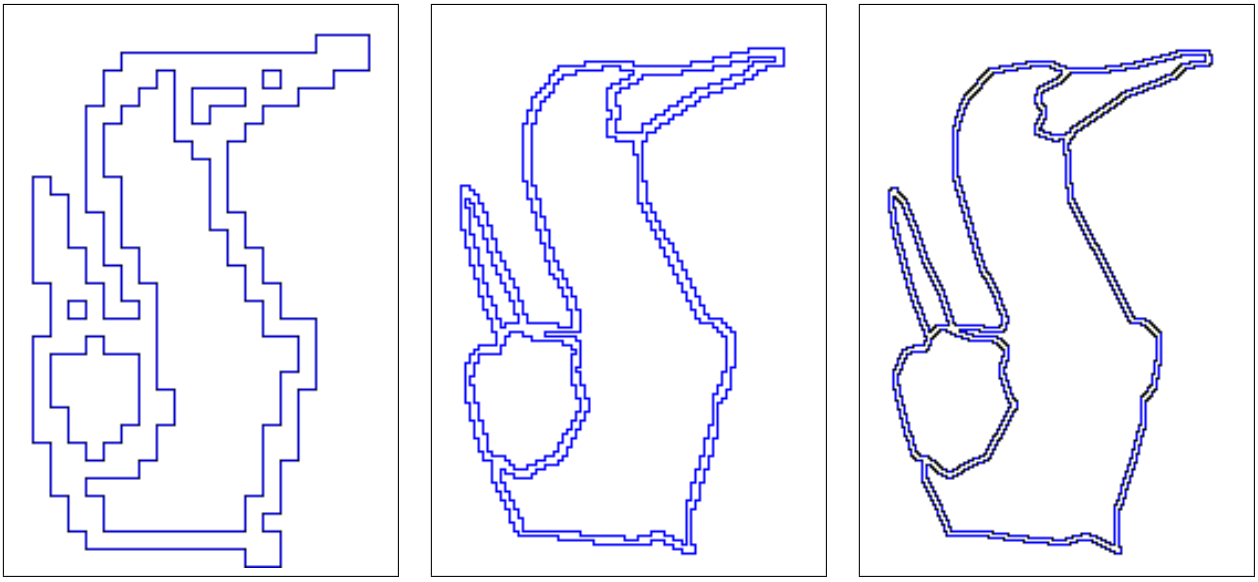
(c) The set of CSS's extracted in stage II of the algorithm **PACE** from the envelope $\mathcal{E}(\mathcal{C}, \mathcal{G})$ shown in (b). The CSS's have been alternately colored in blue and green with the terminal cell of each CSS shown in red.



(d) Final polygonal approximation (in thin black lines) superimposed on the (faded) cellular envelope $\mathcal{E}(\mathcal{C}, \mathcal{G})$.

Figure 8: Results of algorithm **PACE** for cell size $g = 4$ on "duck".

(Fig. 8) — a fact that certifies the desired behavior of a polygonal-approximation algorithm [8, 35]. For $g = 2$, the quality of approximation (measured in terms of the deviation of an approximate polygon from the original curve) improves relative to that corresponding to $g = 4$; and for $g = 1$, the quality of an approximate polygon improves further. However, as evident from Fig. 9, the number of vertices ($M(g = 2) = 45, M(g = 1) = 63$) of the approximate polygon also increases with its improving quality.



Cellular envelopes of the set of thick curves (Fig. 11) representing the image "duck".

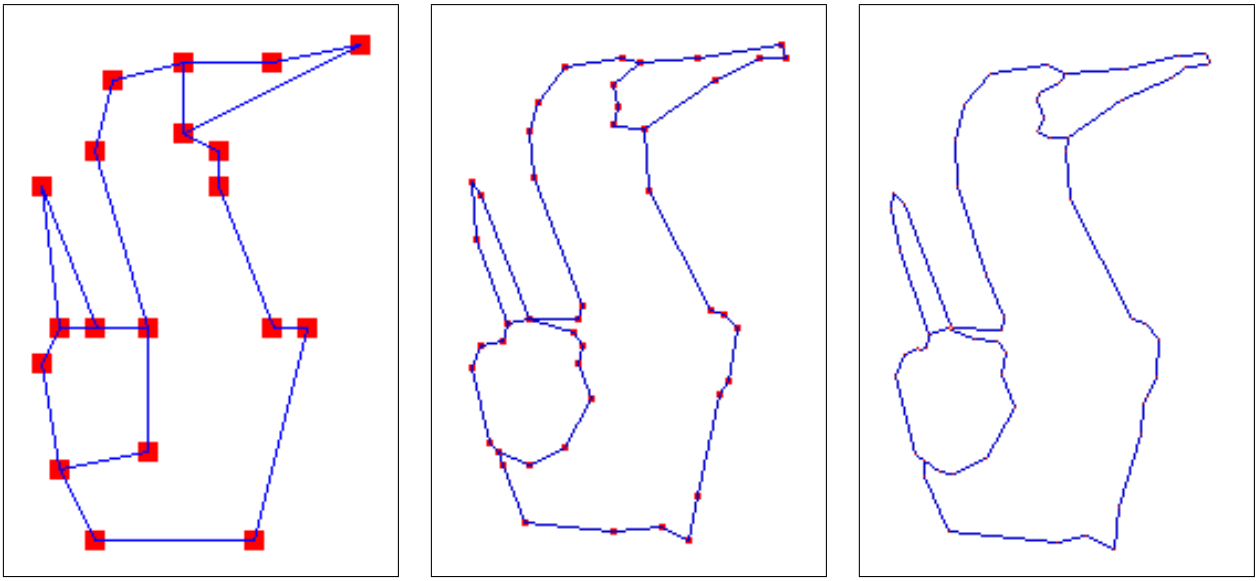


Figure 9: Polygonal approximation of the image "duck" for few other grid sizes ($g = 8, 2, 1$ from left to right) shows how the quality of the output polygon goes on improving with decreasing grid size, although at the cost of increasing number of vertices of the polygon. Note that the vertices of a polygon are highlighted as a red square, the square-size being g, and the edges are shown in blue.

Result-wise comparisons of the algorithm **PACE** with two well-known algorithms, namely **AD** based on *area deviation* [42] and **CM** based on *curvature maxima* [41], are presented in Fig. 11 and in Table 1. Since these algorithms need irreducible/thinned digital curves as input, a thinning algorithm [33] is applied on the set of thick curves constituting the image "duck" as shown in the figure. The set of thinned curves are then

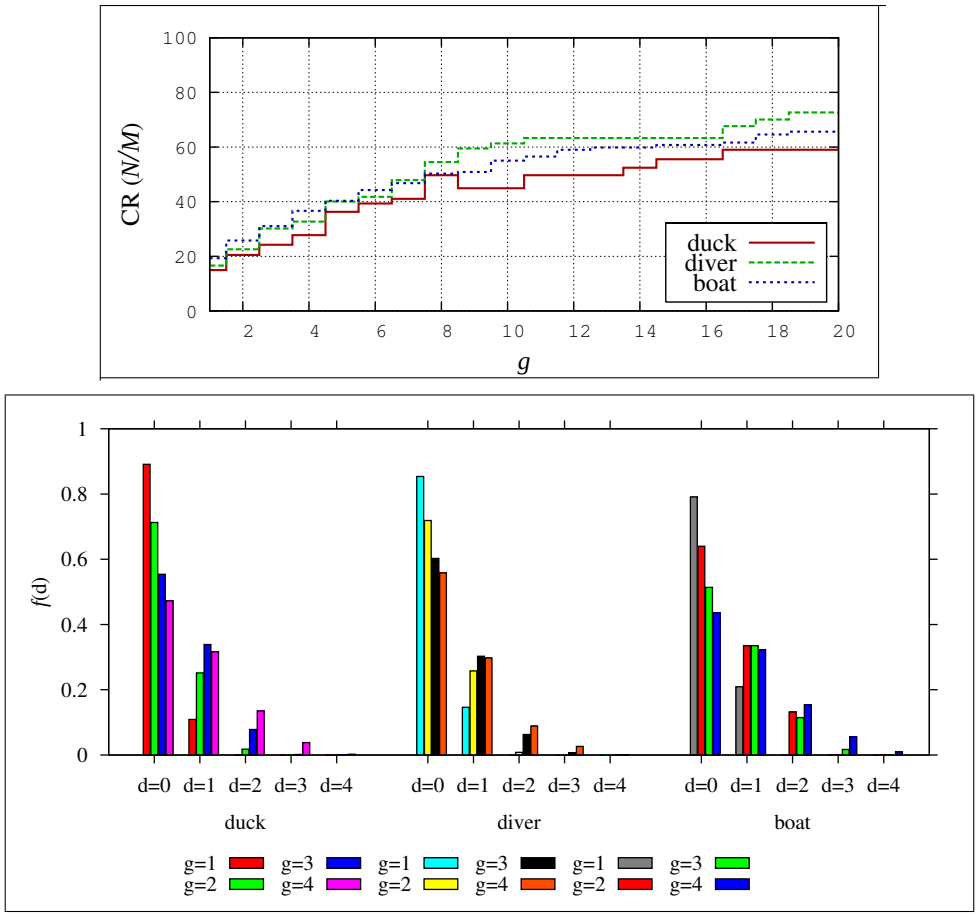
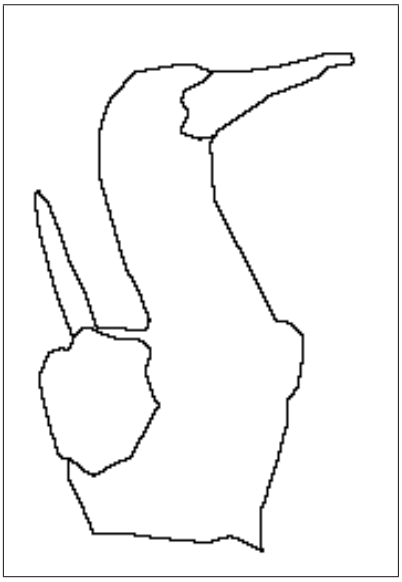


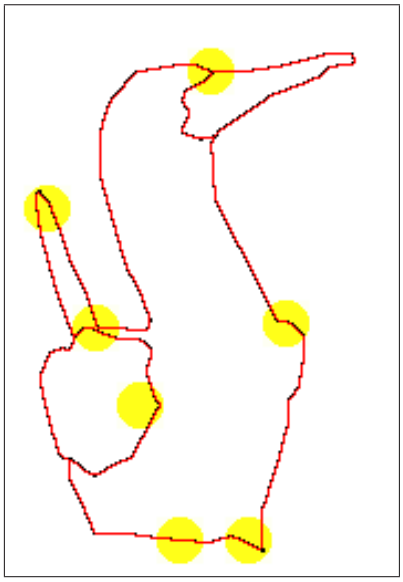
Figure 10: Plots on quality of approximation for the images "duck", "diver", and "boat". The top set of plots shows three histogram profiles on the compression ratio, $CR = N/M$, versus the approximation parameter/cell size, g. The bottom set is made of the histograms on the error frequency $f(d_\perp)$ versus the error of approximation, $d_\perp$, for $g = 1, 2, 4$.

considered as input to these algorithms. It is evident from this figure and Fig. 9 that for a high value of g (e.g., 4), the approximate polygons produced by the algorithm **PACE** are qualitatively inferior to those by area deviation and by curvature maxima. However, the quality of the approximate polygons by the proposed algorithm improves on lowering the value of g. In particular, for $g = 1$ the resultant polygons are qualitatively comparable with the ones by the two other algorithms, as evident from Fig. 11.

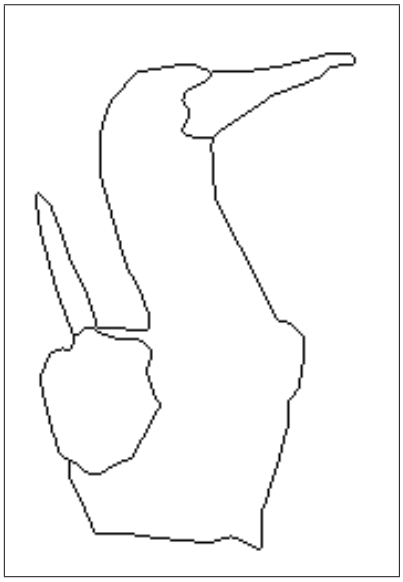
The efficiency of approximation is measured in terms of the compression ratio and the error frequency as explained in Sec. 4.3. The related plots in the form of two sets of histograms for the image "duck" and two other real-world thick curve sets, namely the images "diver" and "boat", are shown in Fig. 10. The respective results of polygonal approximation on "diver" and "boat" are presented in Fig. 13 and Fig. 14 to demonstrate the goodness and efficiency of the algorithm in case of digital curves with varying thickness and arbitrary intersections. For each of the three curve sets, the trade-off between the compression ratio (CR) and the approximation parameter (i.e., the grid size, g) is evident from the set of plots on CR versus g. The set of histograms on the error frequency $f(d_\perp)$ versus the error/deviation $d_\perp$ corresponding to a digital curve $\mathcal{C}$ depict that majority of the erroneous points (with $d_\perp > 0$) have their errors/deviations in the lower range of $[0, g]$, indicating that the approximate polygon lies close to the original (thinned) curve. A relatively much smaller fraction of these erroneous points have high deviations, i.e., deviations nearing g. For example, the polygonal approximation of the image "duck" for $g = 1$ produces $f(d_\perp = 0) = 89\%$ (no error), $f(d_\perp = 1) = 11\%$, and $f(d_\perp > 1) = 0\%$;



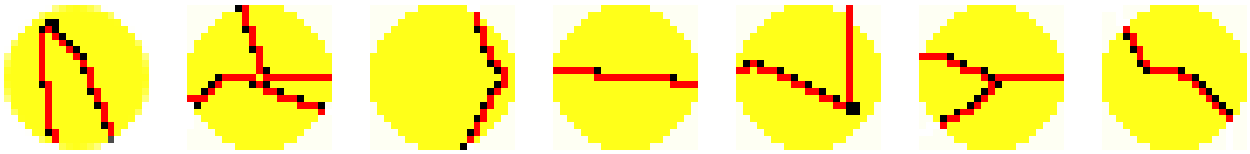
Set of thick curves: Usually a point has more than two points in its 8-neighborhood (8N).



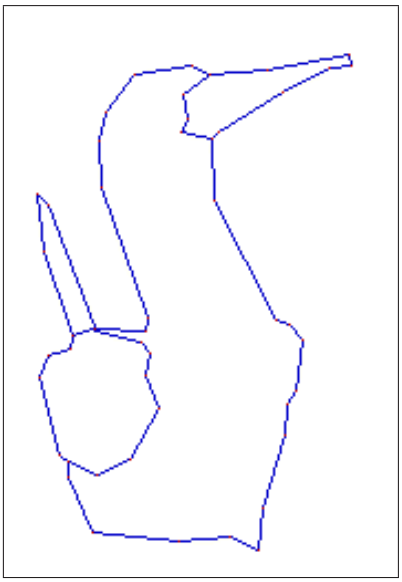
Thinning algorithm: Red points are retained and black points are dropped.



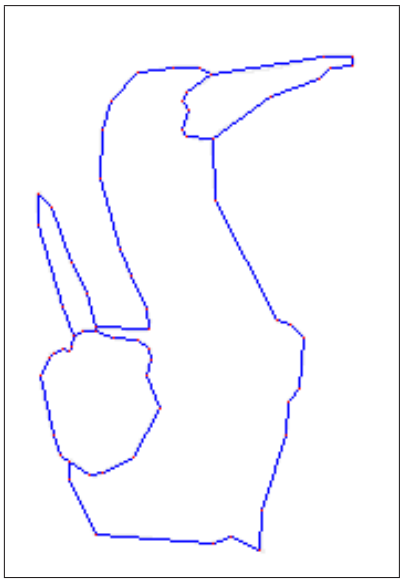
Set of thin curves: A point having more than two points in its 8N indicates a branching.



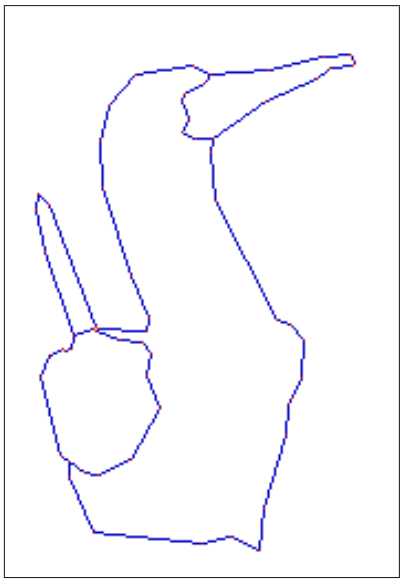
Magnified views of thinning results corresponding to seven portions (colored yellow) of the curve set.



area deviation (Wall & Danielsson [42]): $M = 47$.



curvature maxima (Teh & Chin [41]): $M = 64$.



thinning-free method (proposed): $M(g = 1) = 63$.

Figure 11: Result-wise comparison of the proposed thinning-free method with the existing thinning-based methods for "duck" image.

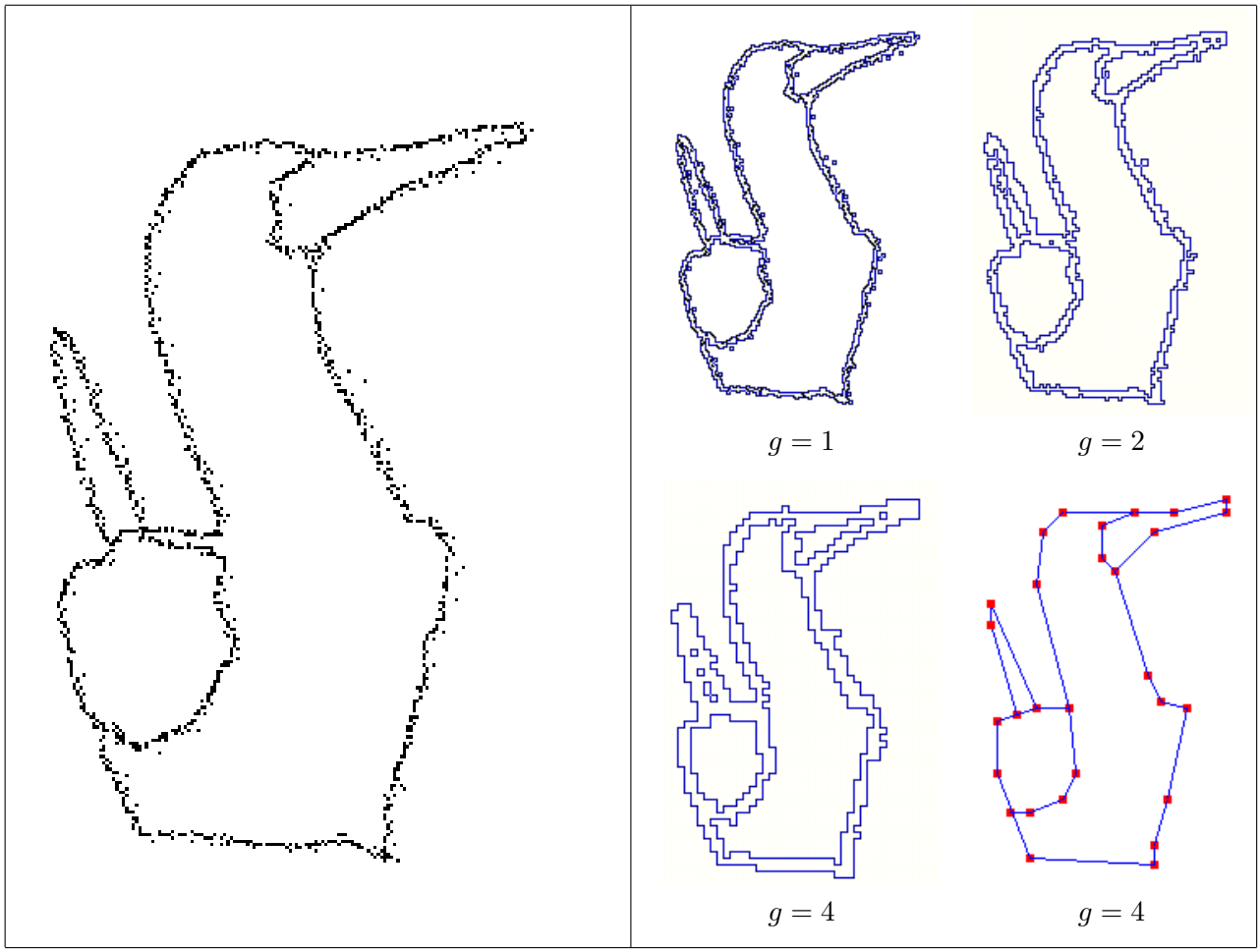


Figure 12: Polygonal approximation of the image "duck" after being injected by salt-and-pepper noise. For a small grid size ($g = 1$ or 2), the resultant cellular envelope cannot cover the underlying object, which is no longer a single connected component. For an appropriately large value of g, however, the object lies entirely in the cellular envelope, thereby making it suitable for polygonal approximation by the proposed method.

for $g = 2$, the corresponding error frequencies are $f(d_\perp = 0) = 72\%$, $f(d_\perp = 1) = 26\%$, $f(d_\perp = 2) = 2\%$, and $f(d_\perp > 2) = 0\%$; and so forth (Fig. 10). More importantly, this is true for each of the three images, which characterizes the independency of the algorithm with the structure and composition of the set of input curves.

One of the salient features of the algorithm **PACE** is its ability to produce effective polygonal approximation even for a noisy digital curve, which, being affected by noise, are likely to be disconnected. A noisy curve can be thought of as a pattern of points located densely along a curve, which are usually disconnected at irregular intervals, although giving the impression of a curve-like pattern due to distribution of the constituting points on or near the actual (noise-free) curve. The existing algorithms of polygonal approximation do not have the provision of working with such noisy curves as input. The set of noisy curves corresponding to the image "duck" and the results of polygonal approximation on it are shown in Fig. 5. Since, for a small value of g (e.g., $g = 1$ or 2), the resultant polygonal envelopes are likely to possess small polygons, which contain the noisy/spurious points of the curve as evident from the figure, a proper value of g is required for applying the proposed algorithm. For example, for $g = 4$, the corresponding polygonal envelope successfully covers all points of the noisy curve, including the spurious points, which enables the algorithm to derive the polygonal approximation to the desired level of precision.

Another feature of the proposed method is the inherent nature of Euclidean-free metrics and operations

Table 1: Comparison of the proposed algorithm **PACE** with the algorithms **AD** using *area deviation* [42] and **CM** using *curvature maxima* [41].

Image name & size	$\lvert\mathcal{C}\rvert$ = # points		$\lvert P\rvert$ = # vertices					Total CPU time (secs.)				
			AD	**CM**	**PACE**			**AD**[a]	**CM**[a]	**PACE**[b]		
	thick	thin			$g=1$	$g=2$	$g=4$			$g=1$	$g=2$	$g=4$
duck 145×227	**1267**	944	47	64	63	45	34	2.717	4.201	0.493	0.283	0.163
diver 333×224	**1789**	1336	69	82	87	63	46	3.733	5.557	0.552	0.328	0.192
boat 364×133	**3319**	2565	97	114	121	92	69	5.470	7.539	0.892	0.507	0.297
india 245×276	**1471**	1173	116	132	137	96	71	5.882	8.421	0.863	0.541	0.300
pyramid 480×198	**2361**	1795	63	67	74	57	43	3.980	4.955	0.672	0.370	0.285

[a]Total CPU time includes the time required for thinning the original curve plus the time required for polygonal approximation of the thinned curve.
[b]Total CPU time includes the time for computing the cellular envelope plus the time required for polygonal approximation the cellular envelope.

involved in both the stages. This imparts high execution speed to the implementation of **PACE**, which is reflected in the respective CPU times shown in Table 1. As evident from this table, the algorithms **AD** and **CM** are considerably slower than **PACE** owing to the usage of only primitive integer operations (comparison, addition, and increment) in the latter. On the contrary, the former algorithms use more complex operations, e.g., multiplication and division, which needs computation in the real domain, for realization of certain geometric and trigonometric concepts as explained in Sec. 1.1. Further, with increase in the cell size g, the number of output vertices of **PACE** decreases significantly. As a result, the compression ratio (CR) improves consistently, but the quality of approximation deteriorates, as evidenced by the quality measures in Fig. 10. As stated earlier, this indicates that the cell size g should be suitably chosen to get an acceptable tradeoff in the approximation.

6 Concluding Remarks

We have presented here the novel concept of approximating a curve-shaped digital object by a cellular envelope. The algorithm is marked by its **(i)** indifference to change in thickness of the input DC, **(ii)** innovative combinatorial approach to construct the optimum cellular envelope for the given DC, **(iii)** use of straightness properties inherited from digital geometry, **(iv)** independency to Euclidean paradigm, and **(v)** realization without any floating point operation, which collectively make it robust, speedy, and efficient.

Although the cellular envelope does not remain entirely unaltered with a different registration of the curve $\mathcal{C}$ w.r.t. the grid (translation), the cellular polygon produced by the subsequent CSS extraction process remains almost invariant. Experimenting on the nature of variation of the cellular envelope and the resulting polygon of a DC with its registration (both translation and rotation) w.r.t. the underlying grid, therefore, stands a possible extension of this work. Placement of a DC with proper orientation w.r.t. the grid in order to obtain its optimal cellular envelope is really a challenging problem, which is not yet addressed. The anisotropic nature of the digital plane, and the apparently unpredictable behavior of the cellular envelope for a shifting/rotating object, calls for innovative digital-geometric techniques to solve this problem.

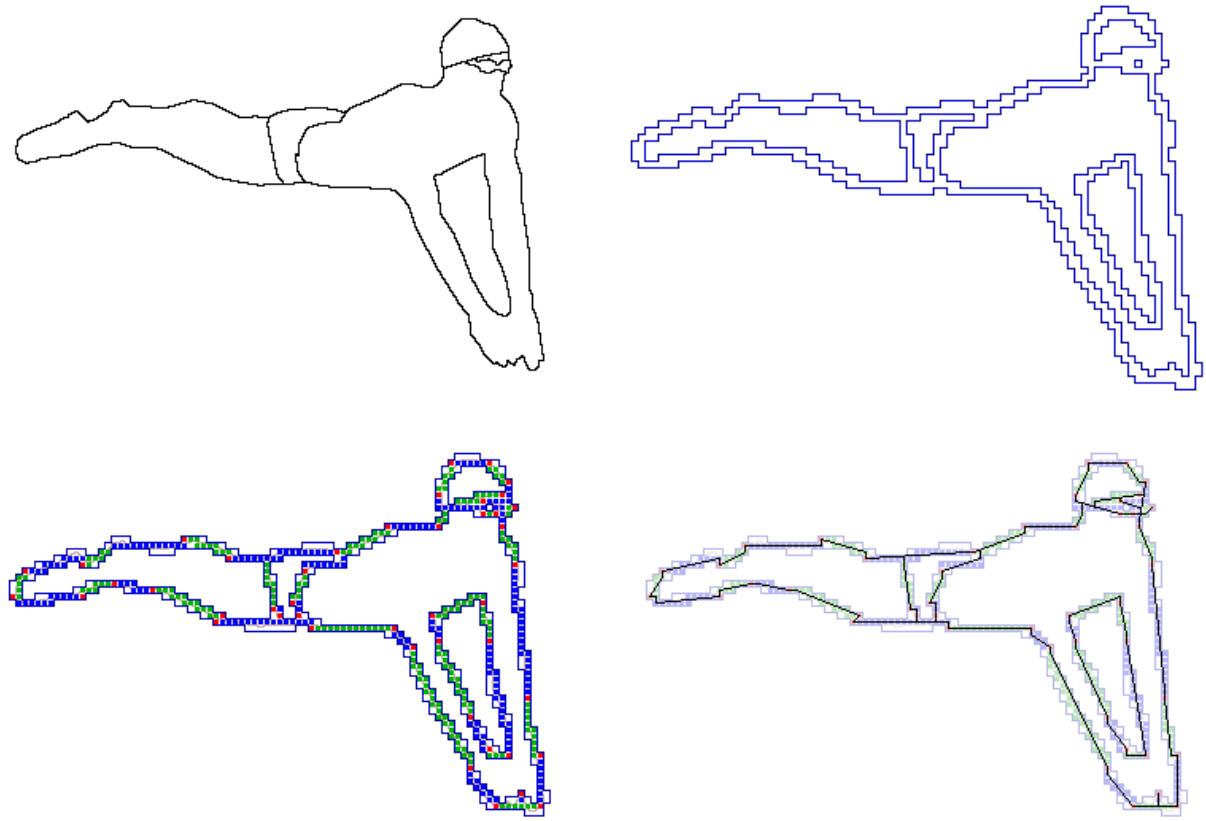


Figure 13: Results on "diver" for $g = 4$. See text and Fig. 8 for explanation.

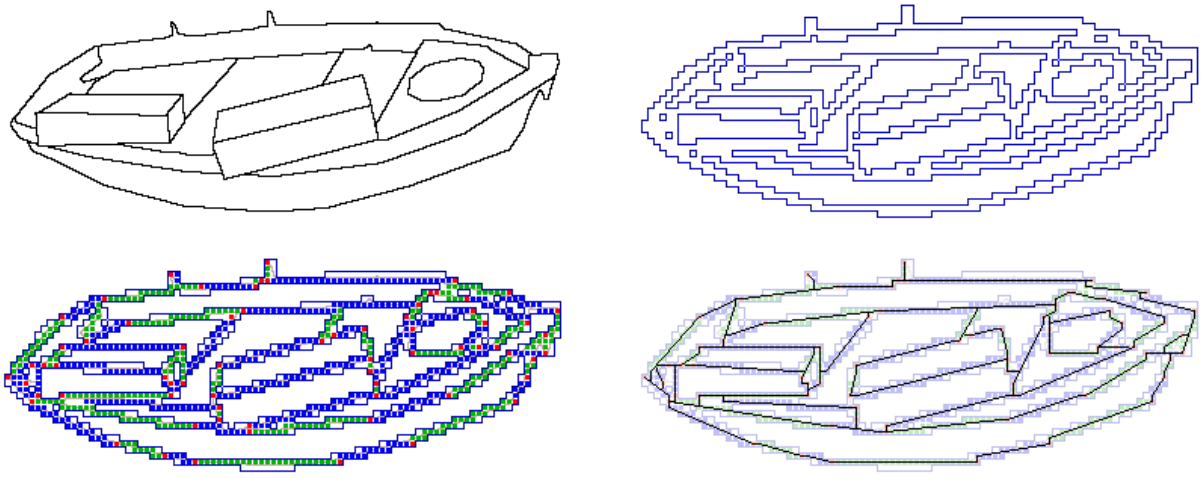


Figure 14: Results on "boat" for $g = 4$. See text and Fig. 8 for explanation.

References

[1] J. R. V. Aken and M. Novak. Curve-drawing algorithms for raster display. *ACM Trans. Graphics*, 4(2):147–169, 1985.

[2] I. M. Anderson and J. C. Bezdek. Curvature and tangential deflection of discrete arcs: A theory based on the commutator of scatter matrix pairs and its application to vertex detection in planar shape data. *IEEE Trans. PAMI*, 6:27–40, 1984.

[3] T. Asano and Y. Kawamura. Algorithmic considerations on the computational complexities of digital line extraction problem. *Systems and Computers in Japan*, 31(14):29–37, 2000.

[4] T. Asano, Y. Kawamura, R. Klette, and K. Obokkata. Digital curve approximation with length evaluation. *IEICE Trans. Fundamentals of Electronics, Communication and Computer Sciences*, E86-A(5):987–994, 2003.

[5] F. Attneave. Some informational aspects of visual perception. *Psychological Review*, 61(3):183–193, 1954.

[6] J. C. Bezdek and I. M. Anderson. An application of the c-varieties clustering algorithms to polygonal curve fitting. *IEEE Trans. Sys., Man & Cybern.*, 15:637–641, 1985.

[7] P. Bhattacharya and A. Rosenfeld. Contour codes of isothetic polygons. *CVGIP*, 50(3):353–363, 1990.

[8] P. Bhowmick and B. B. Bhattacharya. Fast polygonal approximation of digital curves using relaxed straightness properties. *IEEE Trans. PAMI*, 29(9):1590–1602, 2007.

[9] P. Bhowmick, A. Biswas, and B. B. Bhattacharya. Isothetic polygons of a 2D object on generalized grid. In *Proc. 1st Intl. Conf. Pattern Recognition and Machine Intelligence (PReMI)*, volume 3776 of *LNCS*, pages 407–412. Springer, Berlin, 2005.

[10] P. Bhowmick, A. Biswas, and B. B. Bhattacharya. PACE: Polygonal Approximation of thick digital curves using Cellular Envelope. In *5th Indian Conf. Computer Vision, Graphics and Image Processing (ICVGIP)*, volume 4338 of *LNCS*, pages 299–310. Springer, Berlin, 2006.

[11] A. Biswas, P. Bhowmick, and B. B. Bhattacharya. **TIPS**: On finding a **T**ight **I**sothetic **P**olygonal **S**hape covering a 2D object. In *Proc. 14th Scandinavian Conf. Image Analysis (SCIA)*, volume 3540 of *LNCS*, pages 930–939. Springer, Berlin, 2005.

[12] T. C. Chen and K. L. Chung. A new randomized algorithm for detecting lines. *Real Time Imaging*, 7:473–481, 2001.

[13] S. Climer and S. K. Bhatia. Local lines: A linear time line detector. *Pattern Recognition Letters*, 24:2291–2300, 2003.

[14] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. Prentice Hall of India, New-Delhi, 2000.

[15] J. G. Dunham. Optimum uniform piecewise linear approximation of planar curves. *IEEE Trans. PAMI*, 8:67–75, 1986.

[16] A. Fam and J. Sklansky. Cellularly straight images and the hausdorff metric. In *Proc. Conf. on Pattern Recognition and Image Processing*, pages 242–247, 1977.

[17] M. A. Fischler and H. C. Wolf. Locating perceptually salient points on planar curves. *IEEE Trans. PAMI*, 16(2):113–129, 1994.

[18] H. Freeman. On the encoding of arbitrary geometric configurations. *IRE Trans. Electronic Computers*, EC-10:260–268, 1961.

[19] H. Freeman. Techniques for the digital computer analysis of chain-encoded arbitrary plane curves. In *Proc. National Electronics Conf.*, volume 17, pages 421–432, 1961.

[20] H. Freeman and L. S. Davis. A corner finding algorithm for chain-coded curves. *IEEE Trans. Computers*, 26:297–303, 1977.

[21] P. Geer and H. W. McLaughlin. Cellular lines: An introduction. *Discrete Mathematics and Theoretical Computer Science*, pages 167–178, 2003.

[22] D. S. Guru, B. H. Shekar, and P. Nagabhushan. A simple and robust line detection algorithm based on small eigenvalue analysis. *Pattern Recognition Letters*, 25:1–13, 2004.

[23] A. Held, K. Abe, and C. Arcelli. Towards a hierarchical contour description via dominant point detection. *IEEE Trans. Sys., Man & Cybern.*, 24:942–949, 1994.

[24] J. Hu and H. Yan. Polygonal approximation of digital curves based on the principles of perceptual organization. *Pattern Recognition*, 30(5):701–718, 1997.

[25] H. Imai and M. Iri. Computational geometric methods for polygonal approximations of a curve. *Computer Vision, Graphics, and Image Processing*, 36:31–41, 1986.

[26] C. E. Kim. On cellular straight line segments. *Computer Graphics Image Processing*, 18:369–391, 1982.

[27] R. Klette. Cell complexes through time. In *Proc. Vision Geometry*, volume IX of *SPIE 4117*, pages 134–145, 2000.

[28] R. Klette and A. Rosenfeld. *Digital Geometry: Geometric Methods for Digital Picture Analysis*. Morgan Kaufmann Series in Computer Graphics and Geometric Modeling. Morgan Kaufmann, San Francisco, 2004.

[29] R. Klette and A. Rosenfeld. Digital straightness: A review. *Discrete Applied Mathematics*, 139(1-3):197–230, 2004.

[30] T. Pavlidis. Algorithms for shape analysis and waveforms. *IEEE Trans. PAMI*, 2:301–312, 1980.

[31] J. C. Perez and E. Vidal. Optimum polygonal approximation of digitized curves. *Pattern Recognition Letters*, 15:743–750, 1994.

[32] A. Rosenfeld. Digital straight line segments. *IEEE Trans. Computers*, 23(12):1264–1268, 1974.

[33] A. Rosenfeld and A. C. Kak. *Digital Picture Processing, 2nd ed.* Academic Press, New York, 1982.

[34] A. Rosenfeld and R. Klette. Digital straightness. *Electronic Notes in Theoretical Computer Sc.*, 46, 2001. http://www.elsevier.nl/locate/entcs/volume46.html.

[35] P. L. Rosin. Techniques for assessing polygonal approximation of curves. *IEEE Trans. PAMI*, 19(6):659–666, 1997.

[36] P. L. Rosin and G. A. W. West. Non-parametric segmentation of curves into various representations. *IEEE Trans. PAMI*, 17:1140–1153, 1995.

[37] D. Sarkar. A simple algorithm for detection of significant vertices for polygonal approximation of chain-coded curves. *Pattern Recognition Letters*, 14:959–964, 1993.

[38] K. Schröder and P. Laurent. Efficient polygon approximations for shape signatures. In *Proc. Intl. Conf. Image Processing (ICIP),* IEEE CS Press, pages 811–814, 1999.

[39] G. M. Schuster and A. K. Katsaggelos. An optimal polygonal boundary encoding scheme in the rate distortion sense. *IEEE Trans. Circuits and Systems for Video Technology*, 7:13–26, 1998.

[40] S. Tanigawa and N. Katoh. Polygonal curve approximation using grid points with application to a triangular mesh generation with small number of different edge lengths. In *Proc. Intl. Conf. on Algorithmic Aspects in Information and Management, AAIM 2006*, pages 161–172, 2006.

[41] C.-H. Teh and R. T. Chin. On the detection of dominant points on digital curves. *IEEE Trans. PAMI*, 2(8):859–872, 1989.

[42] K. Wall and P.-E. Danielsson. A fast sequential method for polygonal approximation of digitized curves. *Computer Vision, Graphics, and Image Processing*, 28:220–227, 1984.

[43] L.-D. Wu. A piecewise linear approximation based on a statistical model. *IEEE Trans. PAMI*, 6:41–45, 1984.

[44] D. M. Wuescher and K. L. Boyer. Robust contour decomposition using a constant curvature criterion. *IEEE Trans. PAMI*, 13(1):41–51, 1991.

[45] Y. Xie and Q. Ji. Effective line detection with error propagation. In *Proc. Intl. Conf. Image Processing (ICIP),* IEEE CS Press, pages 181–184, 2001.

[46] P.-Y. Yin. A new method for polygonal approximation using genetic algorithms. *Pattern Recognition Letters*, 19(11):1017–1026, 1998.

[47] P. Y. Yin. Ant colony search algorithms for optimal polygonal approximation of plane curves. *Pattern Recognition*, 36:1783–1797, 2003.

[48] P. Y. Yin. A discrete particle swarm algorithm for optimal polygonal approximation of digital curves. *J. Visual Comm. Image Reprsn.*, 15(2):241–260, 2004.

[49] S. Yu and M. Thonnat. Using apparent boundary and convex hull for the shape characterization of foraminifera images. In *Proc. 11th IAPR Intl. Conf. Image, Speech and Signal Analysis*, pages 569–572, 1992.